Week 4: Assignment

1. Which of the following statements is true?
   a. Procedural assignment can be used to update Varilog variables of any type.
   b. The LHS of a procedural assignment can be "{X[3], Y[2:8]}" where X and Y are "reg" type vectors of size 4 and 12 respectively.
   c. The value assigned in a procedural assignment is continuously driven by the expression at the RHS.
   d. Blocking, non-blocking and continuous assignment can be used only inside a procedural block.

   Answer: (b)

   Option (a) is false, as procedural statement can be used to update variables of type reg, integer, real and time. Part of register type variable can be selected for update using procedural statement. Thus option (b) is true. Assigned value remains unchanged until changed by other procedural assignment. Thus option (c) is false. Continuous assignment is not used inside a procedural block. Thus option (d) is also false.

2. For the following code segment, the final value of variable "Z" will be ……

```
integer  X, Y, Z;
initial
      begin
            X = 33; Y = 27; Z = 16;
            #10 Y = X - Z;
            #10 X = Y * 5;
            #10 Z = X + Y;
      end
```

   Answer: 102

   Here the blocking assignment statements will be executed sequentially in the following way:

```
Y = X – Z = 33 – 16 = 17
X = Y * 5 = 17 * 5  = 85
Z = X + Y = 85 + 17 = 102
```

3. For the following code segment, the final value of variable "Z" will be ……

```
integer  X, Y, Z;
initial
      begin
            X = 33; Y = 27; Z = 16;
      end
initial
      begin
            Y <= #10 X - Z;
            X <= #10 Y * 5;
            Z <= #10 X + Y;
```

```
        end
```

Answer: 60

Here all the three non-blocking procedural assignments will be executed parallelly in the following way:

```
Y <= X - Z = 33 - 16 = 17
X <= Y * 5 = 27 * 5  = 135
Z <= X + y = 33 + 27 = 60
```

4. What will the following code segment do?

```
initial
      begin int1=42; int2=17; end
always @(posedge clock)
      int2 = int1;
always @(posedge clock)
      int1 = int2;
```

   a. A race condition will occur, i.e. both the integer variables will update with either value of "int1" or "int2".
   b. Swaps the values of the variables "int1" and "int2".
   c. Both variables will get updated with the value 42.
   d. Both variables will get updated with the value 17.

Answer: (a)

Both the variable int1 and int2 will be updated with the same value of int1 or int2 based on order of execution of the two procedural blocks, which is a race condition. Thus option (a) is true and option (b), (c) and (d) are false.

5. Which of the following statements is not valid?
   a. It is not allowed by simulator to specify blocking and non-blocking statements inside the same always block.
   b. It is not allowed by simulator to specify blocking and non-blocking statements inside the procedural blocks of same module.
   c. It is not allowed to specify the same variable as target of both blocking and non-blocking assignment.
   d. The synthesis process does not consider the delays specified in procedural assignments.

Answer: (a) and (b)

Simulator may allow blocking and non-blocking statements, but it is not a good design practice. Thus option (a) is not valid. Blocking and non-blocking statements may appear inside the procedural blocks of same module. Thus option (b) is also not valid. It is not permitted to use same variable as the target of both blocking and non-blocking assignment. Thus option (c) is valid. Synthesizer ignores the delays specified in the procedural assignment. Thus option (d) is also valid.

6. If the 8-bit variable "x" declared as "`reg [7:0] x`" is initialized to 8'b10101101, what will be its value after execution of the following code segment?

```
always @(posedge clock)
      begin
```

```
            x <= x << 1;
            x[0] <= x[7];
        end
```

a. 8'b01011010
b. 8'b01011011
c. 8'b11010110
d. 8'b01010110

Answer: (b)

Here the value of x is initially 8'b10101101, which after executing of the following non-blocking assignment statements become

x <= x << 1, left shift 1 bit.

Value of x becomes 8'b01011010, and after

x[0] <= x[7], initial value of x[7] which was 1 is assigned to x[0].

Value of x becomes 8'b01011011, left rotation of 1 bit.

7. What will the following code segment generate on synthesis, assuming that the three variables x0, x1 and x2 map into three latches / flip-flops?

```
always @(posedge clock)
    x2 = in;
always @(posedge clock)
    x1 = x2;
always @(posedge clock)
    x0 = x1;
```

a. A 3-bit shift register.
b. Three D flip-flops all fed with the data "in".
c. Values of x0 and x1 will be indeterminate.
d. A 3-bit parallel load register.

Answer: (c)

As all the three blocking assignments will run concurrently inside three always block on the same clock edge, x1 and x0 may get new or old value of x2 and x1 respectively. Thus option (c) is correct whereas option (a), (b) and (d) are not correct.

8. What will the following code segment generate?

```
input clk, in;
output reg [N-1:0] out;
always @(posedge clk)
    begin
        out[0] <= in;
        genvar x;
        generate for (x=1; x<N; x=x+1)
            begin fun_loop
                out[x] <= out[x-1];
            end
    end
```

a. A N-bit shift register will be synthesized.
b. A N-bit arithmetic right shift register will be synthesized..
c. A N-1-bit shift register will be synthesized.

d. All N D flip-flops will be initialized to the value of "in" at every positive "clk" edge.

Answer: (a)

Here the generate block dynamically creates N-1 non-blocking assignment statements where in the LHS of these assignment statements variables x[1], x[2], … , x[N-1] will be updated with the values of variables x[0], x[1], …, x[N-2] respectively and x[0] is assigned new input value of variable "in". Due to the use of non-blocking assignments, all N variables will be updated parallelly on the same positive edge of the clock "clk". Thus option (a) is correct, and options (b), (c) and (d) are not correct.

9. What will the following code segment generate on synthesis?

```
input clk, reset, set;
output reg X;
table
//    clk   reset set    X    X_next
       ?      0     1   : ? :    0;
       ?      1     0   : ? :    1;
       f      1     1   : 1 :    0;
       f      1     1   : 0 :    1;
      (0?)    1     1   : ? :    _;
endtable
```

a. A positive edge-triggered T flip-flop with asynchronous active low set and reset.
b. A negative edge-triggered T flip-flop with asynchronous active low set and reset.
c. A positive edge-triggered D flip-flop with synchronous active low set and reset.
d. A negative edge-triggered D flip-flop with synchronous active low set and reset.

Answer: (b)

Here the UDP describes the state table of a sequential function having the following characteristics:

i. The 1-bit sequential element can be set or reset independent of clock "clk" by keeping the corresponding "set" or "reset" signal active low.
ii. The state of the element toggles on the negative edge of the "clk".
iii. The positive edge of the "clk" is ignored.

Thus option (b) is correct whereas option (a), (c) and (d) are not correct.

10. Which of the following statements are true for user defined primitives in Verilog?
    a. Don't care value is only allowed in specifying input where as no change indicator can only appear as output.
    b. The primitive can be used to specify sequential as well as combination circuit.
    c. The primitive can take as arguments any number of input and output variables.

d. Input and output variables of type scalar and vector both are allowed.

Answer: (a) and (b)
Option (a) and (b) state the design rules of UDP and are true. Since an UPD can take any number of scalar input and only one scalar output, option (c) and (d) are false.